neuflow
Real-Time Vision

Clément Farabet & Yann LeCun

joint work with:
Yann LeCun, Laurent Najman, Marco Scoffier, Srinivas Turaga, Camille Couprie,
Eugenio Culurciello, Berin Martini, Polina Akselrod, Darko Jelaca,
- the model has an **homogenous architecture**: a minimal set of operators is sufficient to compute it
- the model is **intrinsically parallel**: hidden units could be computed by thousands of asynchronous/parallel processors
- inference is **feed-forward**, and therefore deterministic in time
- the model can accommodate very **low numeric precision** (this low precision can be even be accounted for during training)
“the software written for the brain” executes on several billion BPUs*, each connected to several thousand other BPUs.

there is essentially no flow control, the information flows from a BPU to another as needed: no circuitry is wasted in fetching, caching, decoding, and executing code.

* Brain Processing Unit, more commonly known as neuron
**neuFlow** is a dataflow processor architecture running on standard FPGAs

- Reduced flow control
- A grid of dataflow processing units:
  - Each unit is 100% data-driven
  - Data flows across units using efficient data buses
- The grid can be efficiently reconfigured at runtime to unroll algorithms in time
neuFlow: Architecture

A Runtime Reconfigurable Dataflow Architecture

- Configurable Route
- Global Data Lines
- Runtime Config Bus

- multi-port memory controller (DMA)
  [x12 on a V6 LX240T]

- RISC CPU, to reconfigure tiles and data paths, at runtime

- grid of passive processing tiles (PTs)
  [x20 on a Virtex6 LX240T]

- global network-on-chip to allow fast reconfiguration
**neuFlow: Processing Tile (PT) Structure**

- **Term-by-term streaming operators** (MUL, DIV, ADD, SUB, MAX)
- **Configurable bank of FIFOs**, for stream buffering, up to 10kB per PT
- **Full 1/2D parallel convolver** with 100 MAC units
- **Configurable piece-wise linear or quadratic mapper**
- **Configurable router**, to stream data in and out of the tile, to neighbors or DMA ports

[Diagram showing the structure of a Processing Tile (PT) with specific components highlighted]

[Virtex6 LX240T]
luaFlow: A Dataflow Compiler

a home-grown compiler that compiles convolutional networks and the likes to sequences of grid reconfigurations (i.e. neuFlow bytecode)

Joint Work:
C.Farabet
B.Martini
E.Culurciello
Y.LeCun
luaFlow: A Dataflow Compiler

high-level (functional) description

```
net = nn.Sequential()
net:add(nn.SpatialConvolution(3,6,9,9))
net:add(nn.Tanh())
net:add(nn.SpatialSubSampling(6,4,4))
net:add(nn.SpatialConvolution(6,12,9,9))
net:add(nn.SpatialLinear(12,6))
```

Torch7 code
(see our poster)
Infer a flow-graph model from the user description.
luaFlow: A Dataflow Compiler

divide the graph into subgraphs that fit on the grid
for each subgraph, generate the routes and configs for each PT and DMA port

once configured, data streams ripple through the grid,

the grid is “passive”
global optimization: instruction reordering

configuration cycles

data streaming cycles
luaFlow: Supported Operations

Coding: Q8.8 (16bit, fixed-point)

- 1D convolution
- 2D convolution
- local pooling/subsampling/histogramming (max, average, weighted)
- term-by-term div/add/sub/mul/muladd
- point-wise non-linear mapping
- local contrast normalization
- temporal difference
- ...

Monday, December 19, 11
## Performance

<table>
<thead>
<tr>
<th></th>
<th>Intel i7 4cores</th>
<th>nVidia GT335m</th>
<th>neuFlow Virtex 6</th>
<th>nVidia GTX480</th>
<th>neuFlow IBM 45nm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Peak GOP/sec</strong></td>
<td>40?</td>
<td>182</td>
<td>160</td>
<td>1350</td>
<td>1280</td>
</tr>
<tr>
<td><strong>Actual GOP/sec</strong></td>
<td>12</td>
<td>54</td>
<td>147</td>
<td>294</td>
<td>1164</td>
</tr>
<tr>
<td><strong>FPS</strong></td>
<td>14</td>
<td>67</td>
<td>182</td>
<td>374</td>
<td>1480</td>
</tr>
<tr>
<td><strong>Power (W)</strong></td>
<td>50</td>
<td>30</td>
<td>10</td>
<td>250</td>
<td>5</td>
</tr>
<tr>
<td><strong>Embed? (GOP/s/W)</strong></td>
<td>0.24</td>
<td>1.8</td>
<td>14.7</td>
<td>1.176</td>
<td>232.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Float32</th>
<th>Float32</th>
<th>Fixed16</th>
<th>Float32</th>
<th>Fixed16</th>
</tr>
</thead>
</table>

*computing tanh activations after a 16x10x10 filter bank over a 4x500x500 input image*
Application: Scene Parsing
(dense scene-labeling, full scene-understanding)

Joint Work:
C. Farabet
M. Scoffier
C. Couprie
L. Najman
Y. LeCun

goal: dense classification of natural images
Multiscale Feature Learning and Purity Trees

- A multi-scale convolutional network is trained to produce good features for region classification.

- A hierarchy of segmentations is constructed, and a class purity criterion is used to decide if a segment contains a single object.

- An optimal (maximum entropy) cover of the pixels is found to produce the final parse.
Multiscale Feature Learning and Purity Trees

❖ each feature vector represents a local region as well as its context, up to the complete scene

❖ a set of non-disjoint segments are explored to produce the cover

❖ the complete system is a parameter-free task-driven segmentation method

❖ ... and a complete scene can be parsed very efficiently
## Application: Scene Parsing

<table>
<thead>
<tr>
<th>8-class problem</th>
<th>Gould et al.</th>
<th>Munoz et al.</th>
<th>Socher et al.</th>
<th>Kumar et al.</th>
<th>Multiscale ConvNet + SegmTree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixelwise Accuracy (%)</td>
<td>76.4</td>
<td>76.9</td>
<td>78.1</td>
<td>79.4</td>
<td>79.5</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>?</td>
<td>66.2</td>
<td>?</td>
<td>?</td>
<td>74.3</td>
</tr>
<tr>
<td>Inference Time</td>
<td>1 to 10min</td>
<td>10sec</td>
<td>?</td>
<td>1 to 10min</td>
<td>1sec</td>
</tr>
</tbody>
</table>

Results on the Stanford Background Dataset (Gould et al)
### Application: Scene Parsing

<table>
<thead>
<tr>
<th>33-class problem</th>
<th>Liu et al.</th>
<th>Tighe et al.</th>
<th>MCNet+Tree Natural Freqs</th>
<th>MCNet+Tree Freq Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixelwise Accuracy (%)</td>
<td>76.4</td>
<td>76.9</td>
<td>78.5</td>
<td>74.2</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>?</td>
<td>66.2</td>
<td>29.6</td>
<td>46.0</td>
</tr>
</tbody>
</table>

*results on the SiftFlow Dataset (Liu et al)*

<table>
<thead>
<tr>
<th>170-class problem</th>
<th>Tighe et al.</th>
<th>MCNet+Tree Natural Freqs</th>
<th>MCNet+Tree Freq Balancing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixelwise Accuracy (%)</td>
<td>66.9</td>
<td>67.8</td>
<td>39.1</td>
</tr>
<tr>
<td>Per Class Accuracy (%)</td>
<td>7.6</td>
<td>9.5</td>
<td>10.7</td>
</tr>
</tbody>
</table>

*results on the Barcelona Dataset (Tighe et al)*
results on the SiftFlow Dataset (Liu et al)
running on neuFlow: 50ms per frame against 500ms in software (i7,4cores,SSE4)
trained on only 500 images
tested (here) on arbitrary new york scenes
Application: Dense Optical Flow

Joint Work:
M.Scoffier
C.Farabet
Y.LeCun

groundtruth

prediction from our c-to-f convnet

training a multiscale ConvNet to approximate C.Liu’s optical flow

Monday, December 19, 11
once trained, the ConvNet can predict a dense optical-flow at a much faster speed than CRF-based models (simple feedforward inference)
thank you

www.neuflow.org